Introduction to security and cryptography
Lecture 2: Symmetric crypto, block ciphers

Marius Lombard-Platet

ENS, Almerys

Prolegomena

Slides are available on Github:
`https://github.com/mariuslp/iut_crypto/`

Updates will come shortly after each lecture.

Also, please help me with your (anonymous) feedback: `(URL given during lecture)`

News since last lecture

On Friday, Robert E. Grant, from the company 'Crown Sterling' (USA), has presented to the press their revolutionary method for factoring RSA keys.

They factored a 256-bits RSA key in a bit less than a minute. They said they could factor a 512-bits keys in $\sim$5 hours.

Thoughts?

News since last lecture

The crypto community had a good laugh about it. The ones who already knew Crown Sterling past ‘exploits’ just sighed. Longly.

Reminder from last lecture: current recommended keysize for RSA is 2048 bits. Also, 256 bit keys have been factored for more than 20 years.

News since last lecture

The crypto community had a good laugh about it. The ones who already knew Crown Sterling past 'exploits' just sighed. Longly.

Reminder from last lecture: current recommended keysize for RSA is 2048 bits. Also, 256 bit keys have been factored for more than 20 years.

I'm not even going to talk about their confusion between discrete logarithm (ElGamal, Diffie-Hellman...) and factoring (RSA, Paillier...).

Anyway, this is why learning crypto basics is essential to avoid falling into the trap of people who pretend they know what they talk about.

Today

- Block ciphers
- ECB/CBC/CTR... modes

Introduction

Previously

One-time pad achieves perfect secrecy:

Previously

One-time pad achieves perfect secrecy: given a ciphertext, all messages (of the same size) can equally probably the matching cleartext.

One caveat: we need keys as big as the message.

Can we do the same with smaller keys? Or equivalently, can we reuse the keys?

Previously

No.

According to Shannon's information theory (which by the way helped us to define perfect secrecy), we cannot perfectly hide n bits of information with fewer than n bits of randomness.

In fact, even using keys of $n - 1$ bits would leak a non-negligible amount of data.

Perfection is overrated anyway

However, we don't really need *perfect* secrecy. Most of the time, we are fine if breaking the message would require hundreds of years.

Hence **computational security**. Remember λ, the security parameter?

Modulus (bits)	Operations ($\log_2$)
512	58
1024	80
2048	111
4096	149
8192	156

Bits of security

The operation number is called the number of **bits of security**. Today, 112 bits of security are recommended for any cipher. Until 2014, it used to be 80.

The size has increased because computers are faster, hence can do more operations in less time.

Note that the notation is exponential: one more bit of security means **twice** as much computation. So for instance, 16 more bits of security means $2^{16} = 65536$ times longer computation.

Note and Exercise

Note that the number of bits of security is bounded by the secret key size (as we can bruteforce the key). But it can be much lower, see RSA.

Assume a cryptosystem requires 2^{56} operations to be broken. How many bits of security does it offer?

Another cryptosystem requires 2^{100} operations to break the first half of its key, and 2^{100} operations to break the second half. Is it secure according to today standards?

Note and Exercise

Note that the number of bits of security is bounded by the secret key size (as we can bruteforce the key). But it can be much lower, see RSA.

Assume a cryptosystem requires 2^{56} operations to be broken. How many bits of security does it offer?

Another cryptosystem requires 2^{100} operations to break the first half of its key, and 2^{100} operations to break the second half. Is it secure according to today standards? Answer:
$2^{100} + 2^{100} = 2 \times 2^{100} = 2^{101}$. Nope.

Back to symmetric encryption

Hence, we will use small keys, and try to make the encryption as hard to decrypt as possible.

Back to symmetric encryption

Hence, we will use small keys, and try to make the encryption as hard to decrypt as possible.

Obvious attack: **bruteforce** (exhaustive search) on the key. For instance, DES (proposed in 1976) used 56 bit keys, only 7.10^{16} keys. In 1998, Deep Crack bruteforced a DES key in only 56 hours.

Obvious fix: bigger keys. Triple DES uses 168 bits, AES uses 128, 192 or 256 bits.

Shannon's Principle 1949

Confusion
The purpose of confusion is to make the relation between the key and the ciphertext as complex as possible.

Ciphers not offering much confusion (such as Vigenere cipher) are susceptible to frequency analysis.

Diffusion
Diffusion spreads the influence of a single plaintext bit over many ciphertext bits.

Principle
A good cipher design uses Confusion and Diffusion together

Encrypting arbitrary data

We use small keys. What we can do easily is encrypt a fixed size of data.

If we have more data to encrypt, we'll deal with it later.

This chunk size should not be too small (or bruteforcing would be easy), nor too big (then computations would be long).

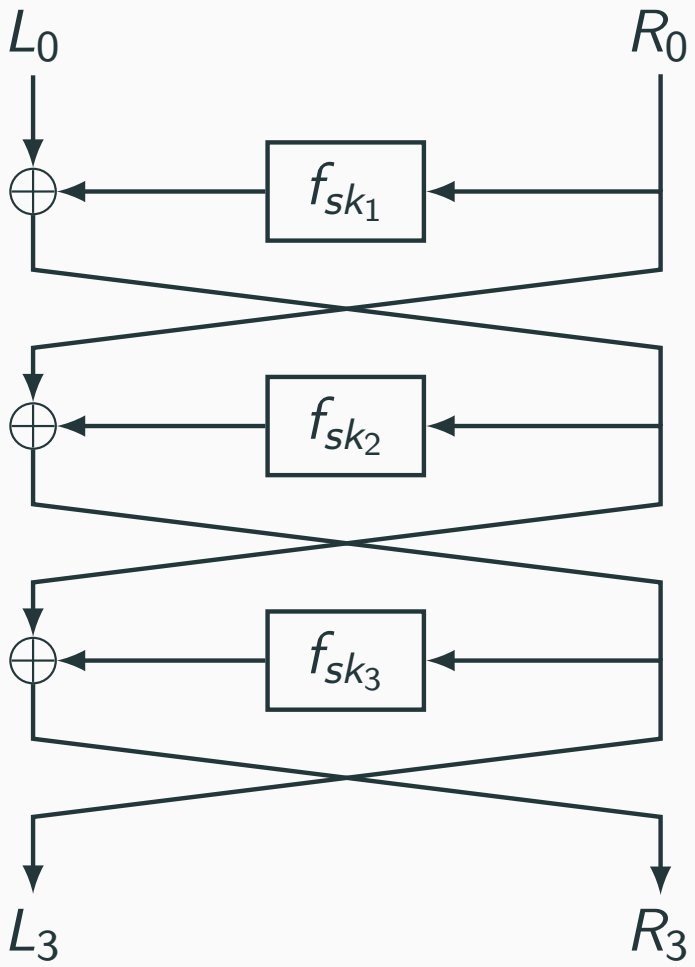
Feistel network

Feistel networks have been invented by IBM engineers, amongst whom Horst Feistel, in the 70s.

- m, message of size n (n even)
- m is split into two blocs of same size, L_0 and R_0, with $|L_0| = |R_0| = \frac{n}{2}$
- $m = L_0 \| R_0$
- A keyed function f is required.

Feistel network

Let's imagine a three rounds Feistel system:



For $1 \leq i \leq 3$:

$L_i = R_{i-1}$

$R_i = L_{i-1} \oplus f(R_{i-1}, sk_i)$

Exercise

- Let $m = 1101111010$.
- $sk_1 = 10101$, $sk_2 = 11001$ and $sk_3 = 10111$.
- $f_{sk}(x) = \bar{x} \oplus sk$

Compute m's corresponding ciphertext with this Feistel network, and these subkeys.

Exercise

How to decrypt a ciphertext from a Feistel network?

Feistel network

Answer: pretty much in the same way.

$$R_{i-1} = L_i \quad L_{i-1} = R_i \oplus f(R_{i-1}, K_i)$$

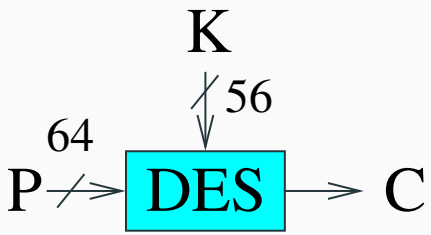
Keys are used in reverse order.

(It is proven that if f is cryptographically secure, then 3 rounds are enough for perfect secrecy. In practice, we use more than that.)

Data Encryption Standard (call in 1973)

Lucifer designed in 1971 by Horst Feistel at IBM. Relies on Feistel network.

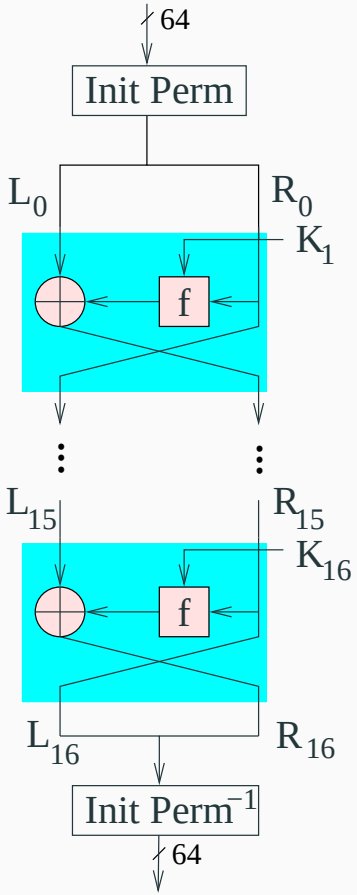
- Block cipher, encrypting 64-bit blocks
  Uses 56 bit keys expressed as 64 bit numbers (8 bits parity checking)



- First cryptographic standard.
  - 1977 US federal standard (US Bureau of Standards)
  - 1981 ANSI private sector standard
  - Ceased to be a standard in 2000

DES — overall form

- 16 rounds Feistel cipher + key-scheduler.
- Key scheduling algorithm derives subkeys K_i from original key K.
- Initial permutation at start, and inverse permutation at end.
- (keyed) f consists of two permutations (P-box) and an s-box substitution.



$L_{i+1} = R_i$ and $R_{i+1} = L_i \oplus f(R_i, K_i)$

Note on permutations

Permutations don't add any security, however they allow a faster computation on hardware using 8-bits bus.

DES (Data Encryption Standard)

What is DES main weakness ?

DES (Data Encryption Standard)

What is DES main weakness ?

The size of the secret key! Only 56 bits. Cracked in 3 days in 1998 by Deep Crack.

In *Symmetric* crypto, it is desirable to have 128 or 256 bit keys.

DES suffers from other vulnerabilities: the existence of *weak keys*, attacks based on linear cryptanalysis, differential cryptanalysis...

2DES

First idea: use 2 DES, with 2 different keys.

Then we have a key of 112 bits.

However, there is a clever attack (meet in the middle) that proves that 2DES is as easy to crack as simple DES.

3DES (Triple DES)

Hence let's go one step further: Triple DES. The meet in the middle attack does not work here. No obvious attack against 3DES.

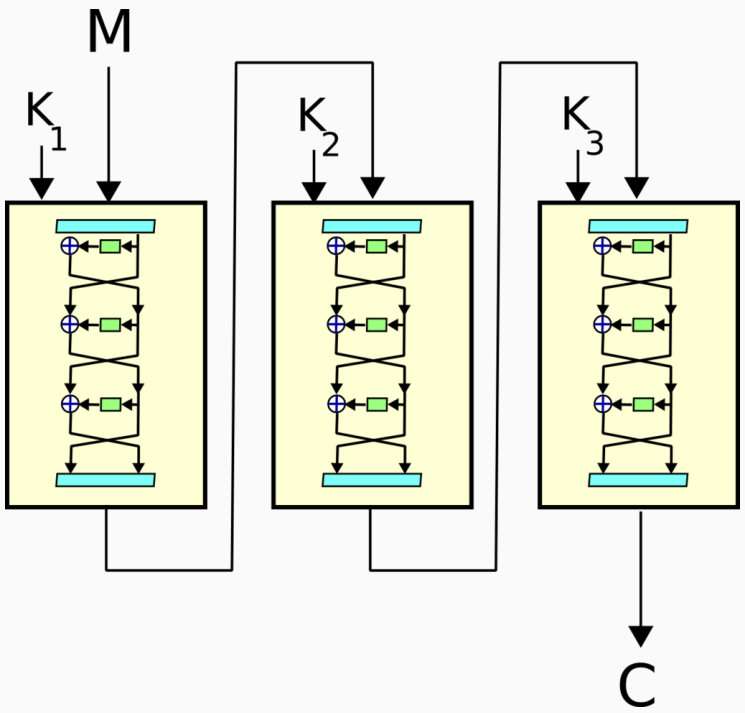
Still used by banking companies, and many many many companies everywhere.

Yet, has been declared obsolete 2 years ago. **Do not use it for new projects!**

3DES relies on 3 secret keys, each for one DES use: sk_1, sk_2 and sk_3.

3DES

We have $C = \mathrm{Enc}_{K_3}(\mathrm{Dec}_{K_2}(\mathrm{Enc}_{K_1}(M)))$



3DES

$$C = \mathrm{Enc}_{K_3}(\mathrm{Dec}_{K_2}(\mathrm{Enc}_{K_1}(M)))$$

Three options

1. Three different keys $\Rightarrow$ one key of 168 bits
2. $sk_1 = sk_3$ and $sk_2 \neq sk_1 \Rightarrow$ one key of 112 bits
3. $sk_1 = sk_2 = sk_3 \Rightarrow$ one key of 56 bits (compatibility with DES)

What is the disadvantage?

3DES

$$C = \mathrm{Enc}_{K_3}(\mathrm{Dec}_{K_2}(\mathrm{Enc}_{K_1}(M)))$$

Three options

1. Three different keys $\Rightarrow$ one key of 168 bits
2. $sk_1 = sk_3$ and $sk_2 \neq sk_1 \Rightarrow$ one key of 112 bits
3. $sk_1 = sk_2 = sk_3 \Rightarrow$ one key of 56 bits (compatibility with DES)

What is the disadvantage?

Three times as slow... And DES was already not super fast.

Find a New Symmetric Scheme

Call to replace DES.

A competition is organized between 1997 and 2001.

AES (Advanced Encryption Standard)

- Block cipher, approved for use by US Government in 2000. Very popular standard, designed by two Belgian cryptographers (Daemen and Rijmen)
- Block-size: 128 bits
- Key size: 128, 192, or 256 bits
- Uses various substitutions and transpositions, and key scheduling in different rounds

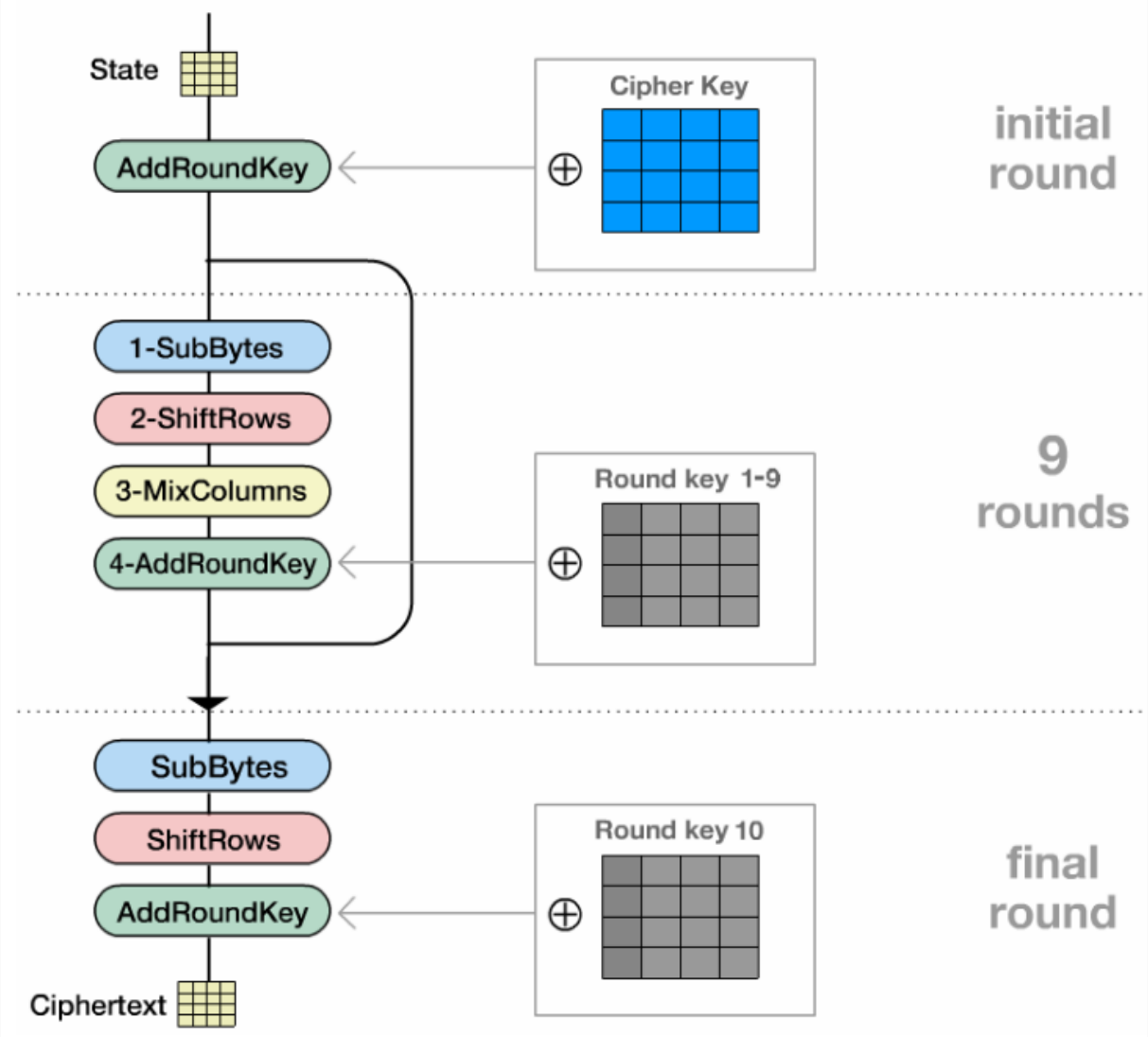
AES (Advanced Encryption Standard)

- Algorithm believed secure. Only attacks are based on side channel analysis, i.e. attacking implementations that inadvertently leak information about the key
- Does not rely on Feistel network, but on SPN (substitution-permutation network)
- Operates on bytes (when DES operates on bits)

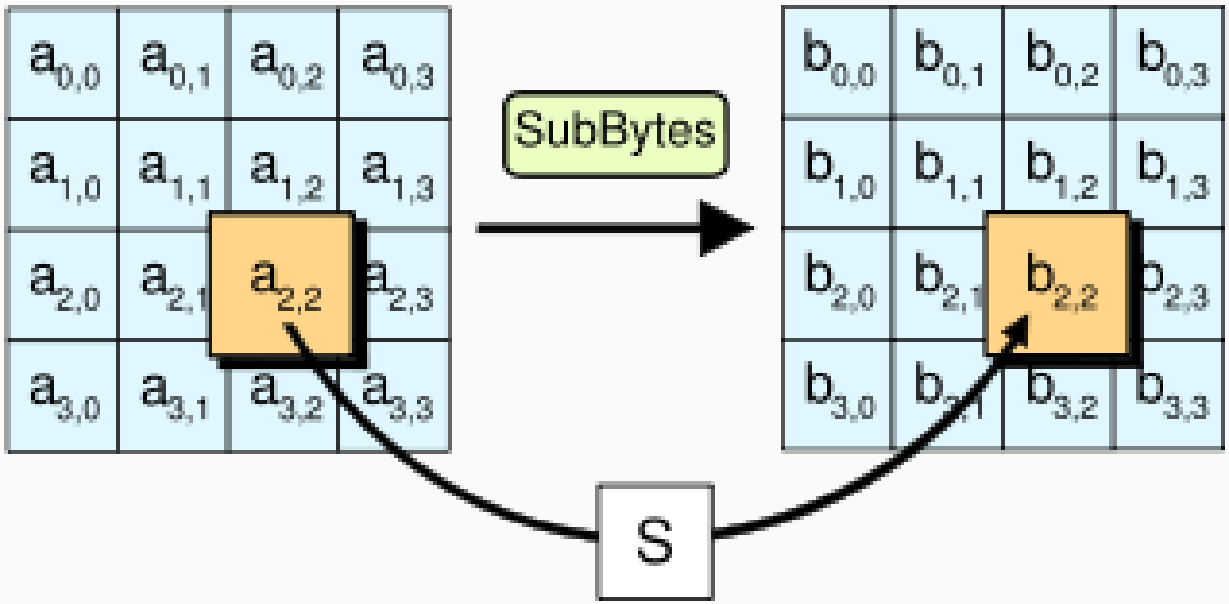
AES (Advanced Encryption Standard)

Key size	# rounds
128	10
192	12
256	14

AES (details are in appendix)



AES / SubBytes (example of S-box)



AES / SubBytes

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

AES / SubBytes

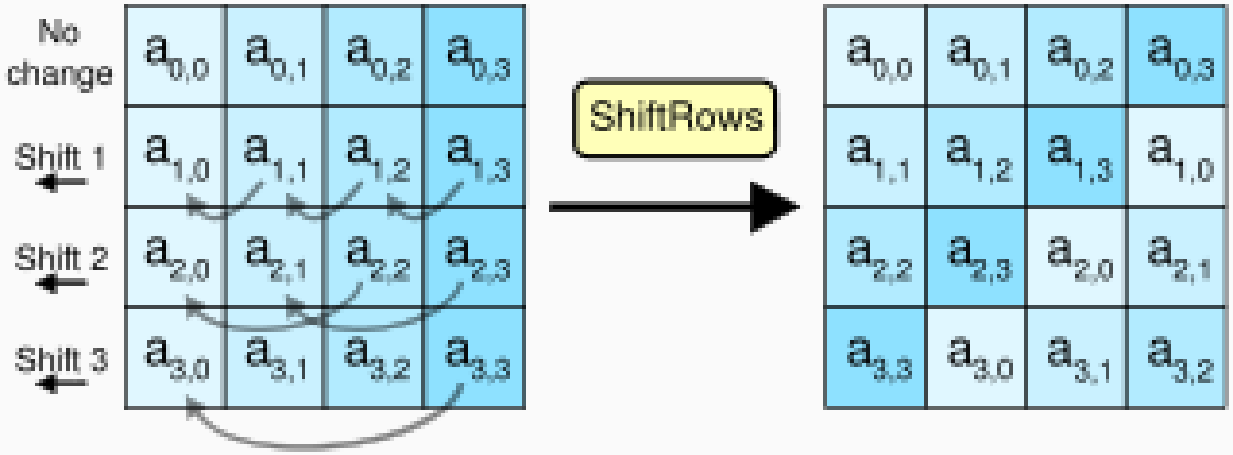
	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Exercise

Apply SubBytes on the following block:

14	f1	3d	02
4d	a5	b7	66
71	9c	81	ac
da	95	6a	e1

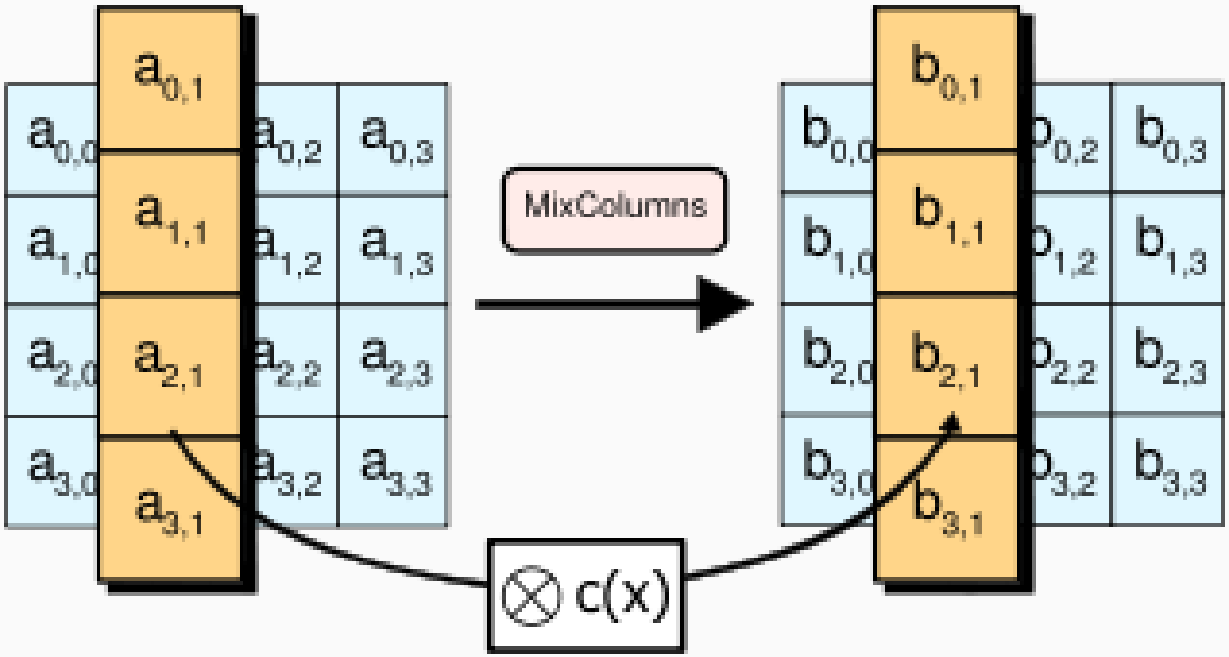
AES / ShiftRows



Exercise
Apply ShiftRows on the following block:

14	f1	3d	02
4d	a5	b7	66
71	9c	81	ac
da	95	6a	e1

AES / MixColumns

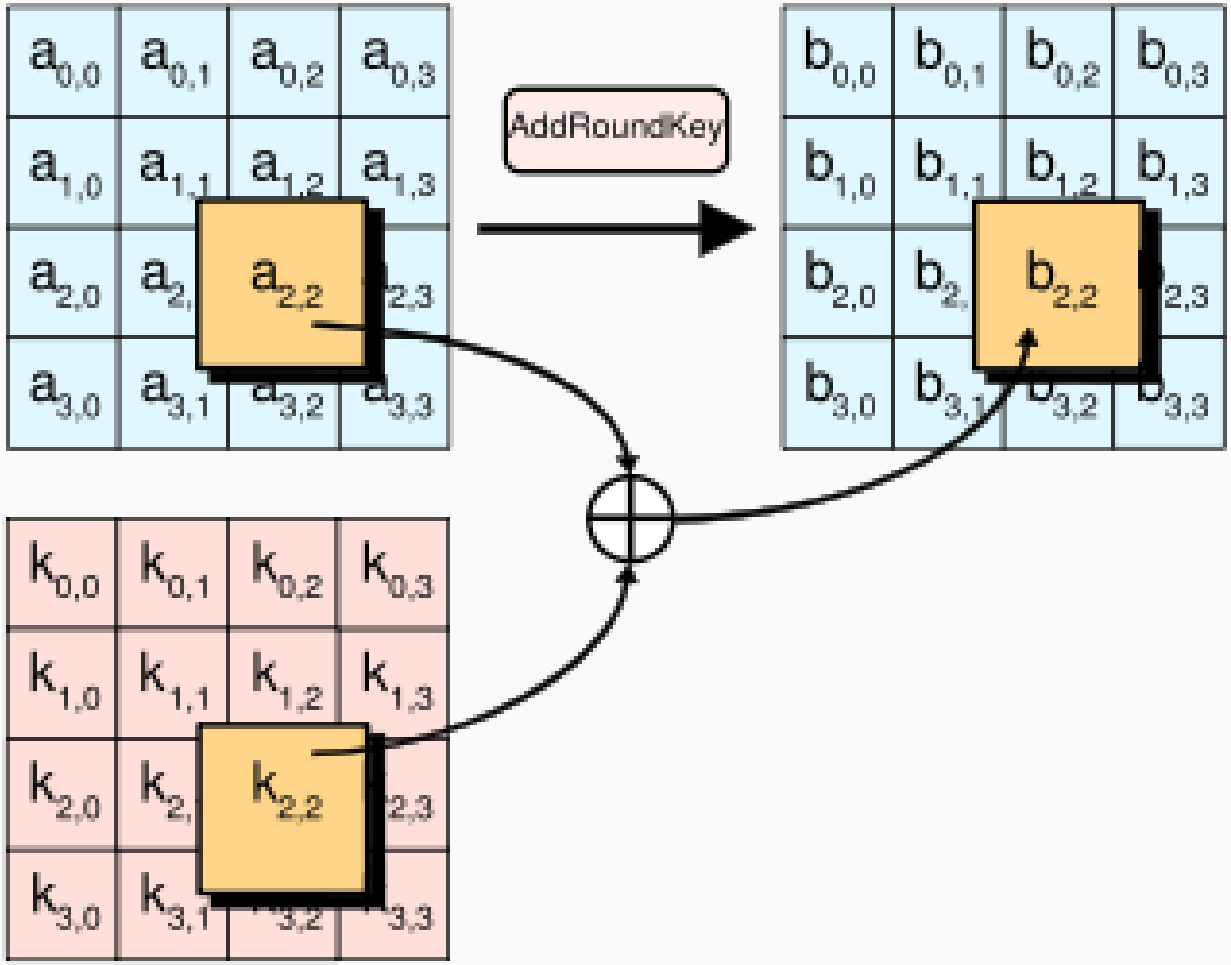


AES / MixColumns

Each column is multiplied by the following matrix:

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

AES / AddRoundKey



Exercise
Add the the round key to the block B:

Round key

14	f1	3d	02
4d	a5	b7	66
71	9c	81	ac
da	95	6a	e1

Block B

23	f1	45	02
6a	a5	e1	1c
4e	c9	ff	5d
b3	01	22	b4

Warning

You may notice that AES operations are not super hard to code, and may be tempted to code you own AES software.

You may do so, but **NEVER USE IT IN REAL LIFE**!

There are many caveats, such as side-channel attacks. For instance, attackers may look at how the algo reacts to errors (fault injection), or to the electric consumption during computation.

In general, never code your own crypto. Use existing libraries.

Other symmetric schemes

Are there other symmetric schemes?

Other symmetric schemes

Are there other symmetric schemes? Yes. And many still in use.

Blowfish, Serpent, Twofish, 3-Way, ABC, Akelarre, Anubis, ARIA, BaseKing, BassOmatic, BATON, BEAR and LION, C2, Camellia, CAST-128, CAST-256, CIKS-1, CIPHERUNICORN-A, CIPHERUNICORN-E, CLEFIA, CMEA, Cobra, COCONUT98, Crab, CRYPTON, CS-Cipher, DEAL, DES-X, DFC, E2, FEAL, FEA-M, FROG, G-DES, GOST, Grand Cru, Hasty Pudding Cipher, Hierocrypt, ICE, IDEA, IDEA NXT, Intel Cascade Cipher, Iraqi, KASUMI, KeeLoq, KHAZAD, Khufu and Khafre, KN-Cipher, Ladder-DES, Libelle, LOKI97, LOKI89/91, Lucifer, M6, M8, MacGuffin, Madryga, MAGENTA, MARS, Mercy, MESH, MISTY1, MMB, MULTI2, MultiSwap, New Data Seal, NewDES, Nimbus, NOEKEON, NUSH, Q, RC2, RC5, RC6, REDOC, Red Pike, S-1, SAFER, SAVILLE, SC2000, SEED, SHACAL, SHARK, Skipjack, SMS4, Spectr-H64, Square, SXAL/MBAL, TEA, Treyfer, UES, Xenon, xmx, XTEA, XXTEA, Zodiac.

Block chaining

Encrypting large data

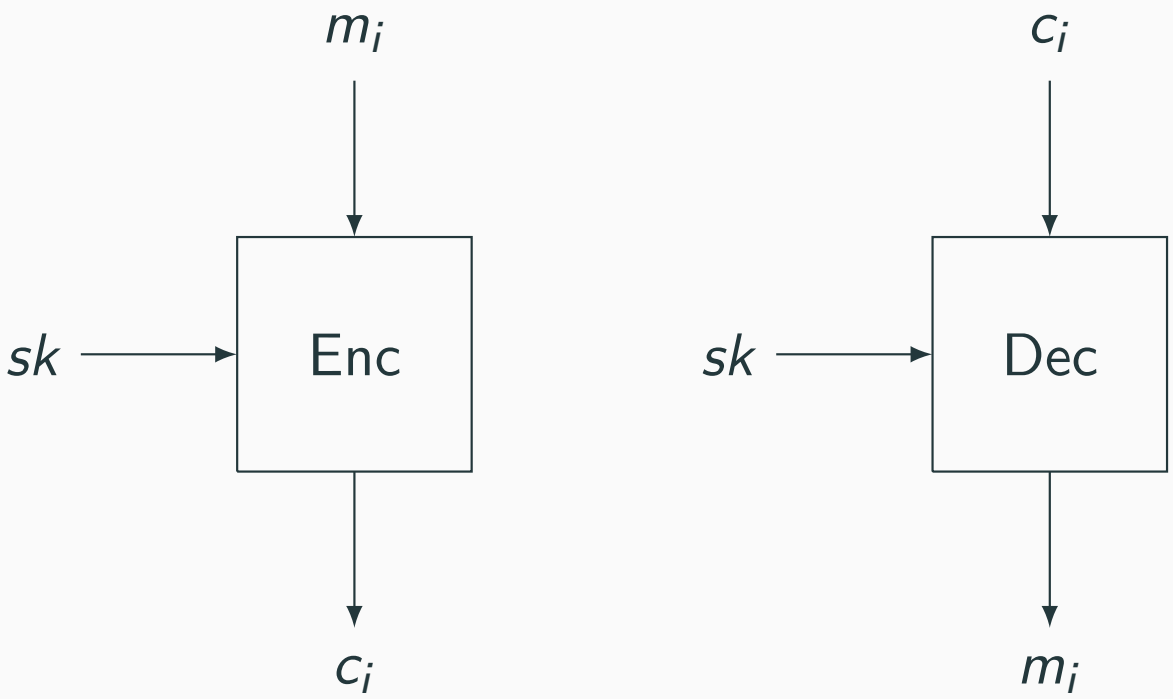
For the moment, we can only encrypt a few bits of data (for AES, we can only encrypt 128 bits)

How to encrypt more with the same key?

Mode ECB (Electronic CodeBook)

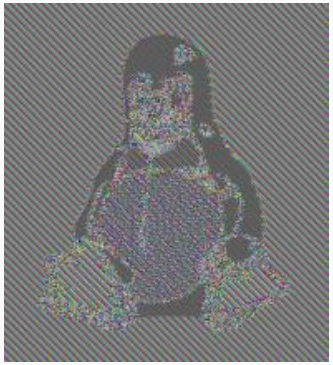
Let $|m| = k \cdot n$ with $k > 1$.

We decompose m as $m = (m_1, \ldots, m_k)$, with $|m_i| = n$ bits.



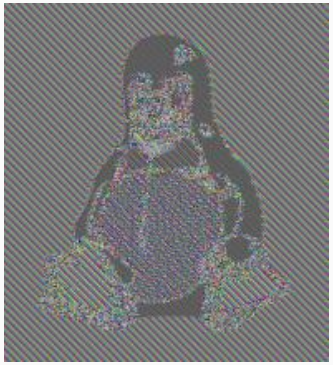
ECB weakness

What we get with ECB mode:

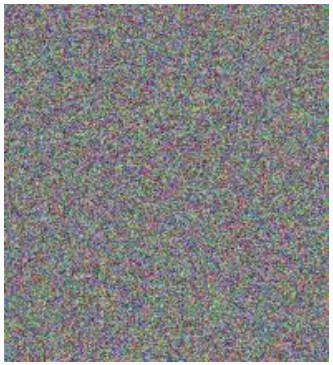


ECB weakness

What we get with ECB mode:



What we'd like (and get with other modes):

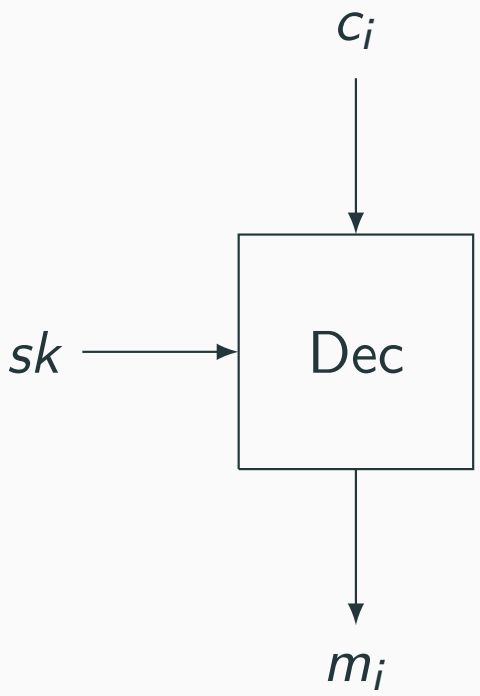


Never use ECB mode.

Exercise

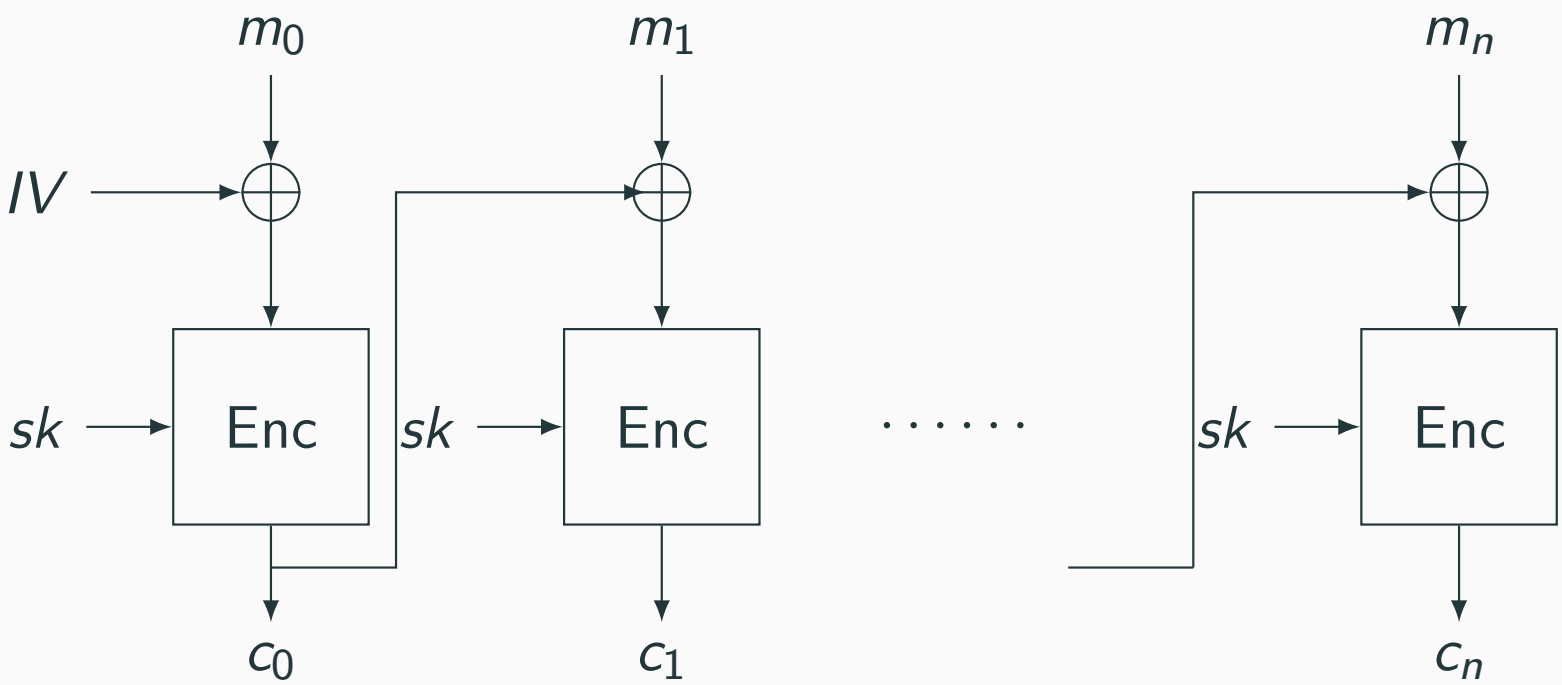
1. What to do if $|m|$ is not a multiple of n?
2. How do you decrypt a ciphertext encrypted in ECB mode?

Answer

1. We can add zeros at the end of the last block, so it reaches 128 bits for AES. This is called **padding**.
2. 

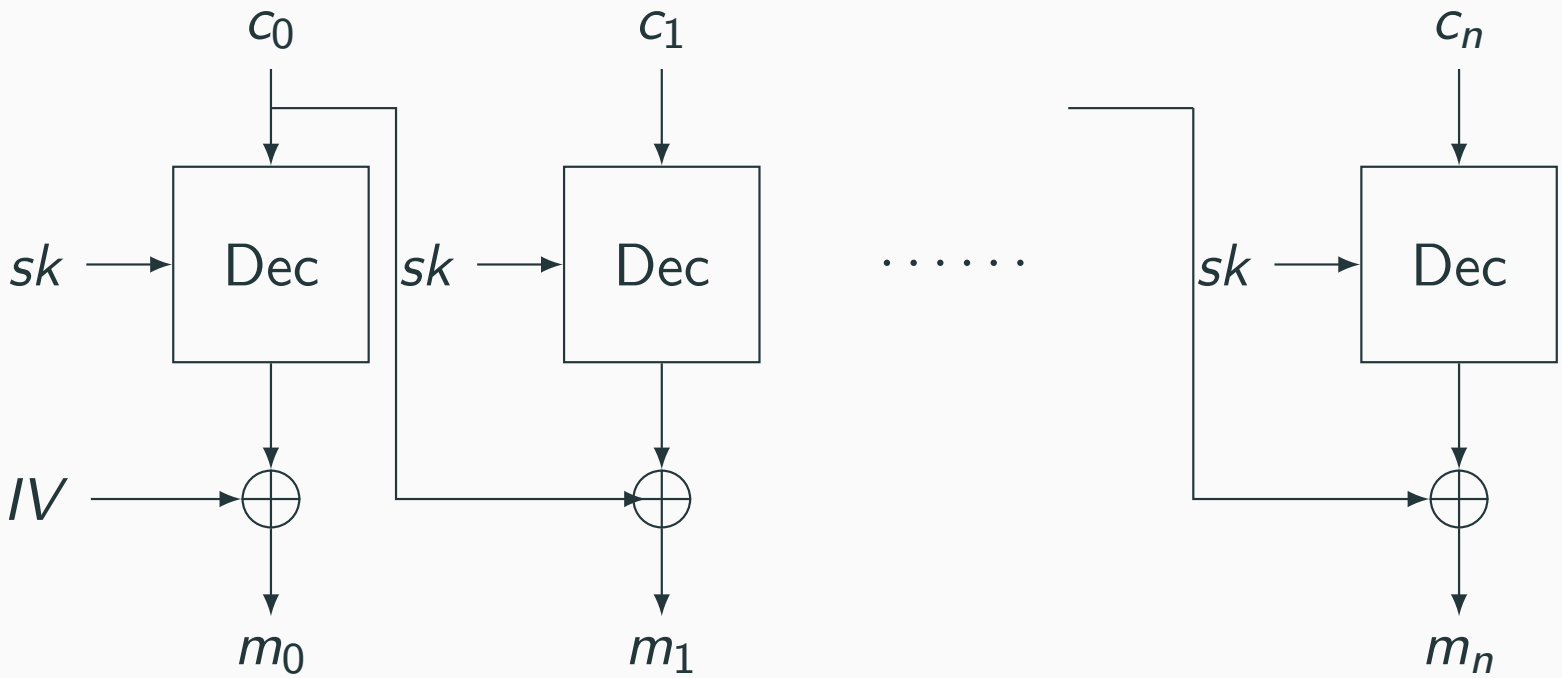
Mode CBC (Cipher Block Chaining)

Encryption:



Mode CBC (Cipher Block Chaining)

Decryption:



Note on CBC

IV is called the initialization vector, and must be the same for encryption and decryption of a same message.

IV need not be secret, however each message must be encrypted with a different IV. This is called a **nonce**.

Making sure IV is unique
Naive solution: store old IVs. Takes place, unconvenient.

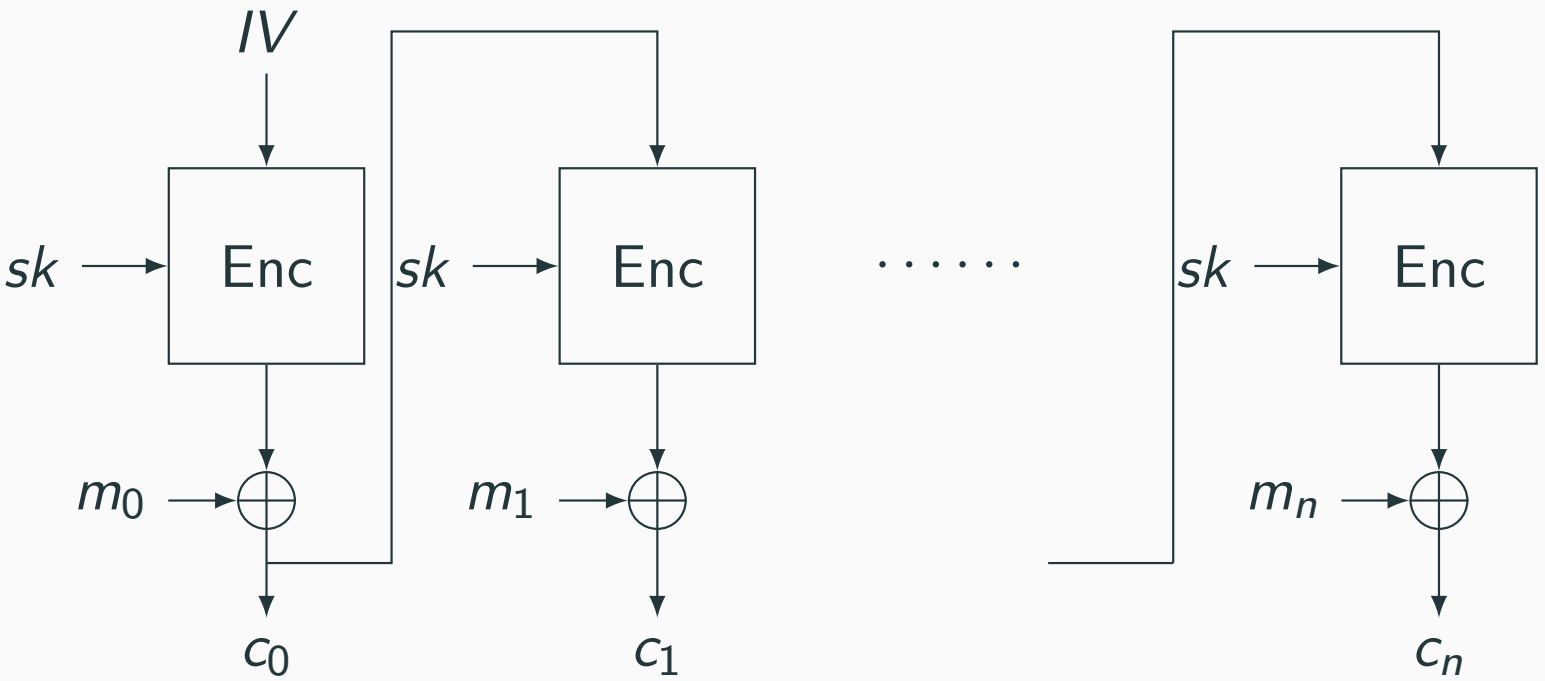
Better solution: IV are big (same size as the blocks). Take randomly one IV. For AES (128 bit blocks), there's a 50% chance of *one* collision after 10^{38} messages. You're fine.

Exercise

- What can the attacker learn if the same IV is used?

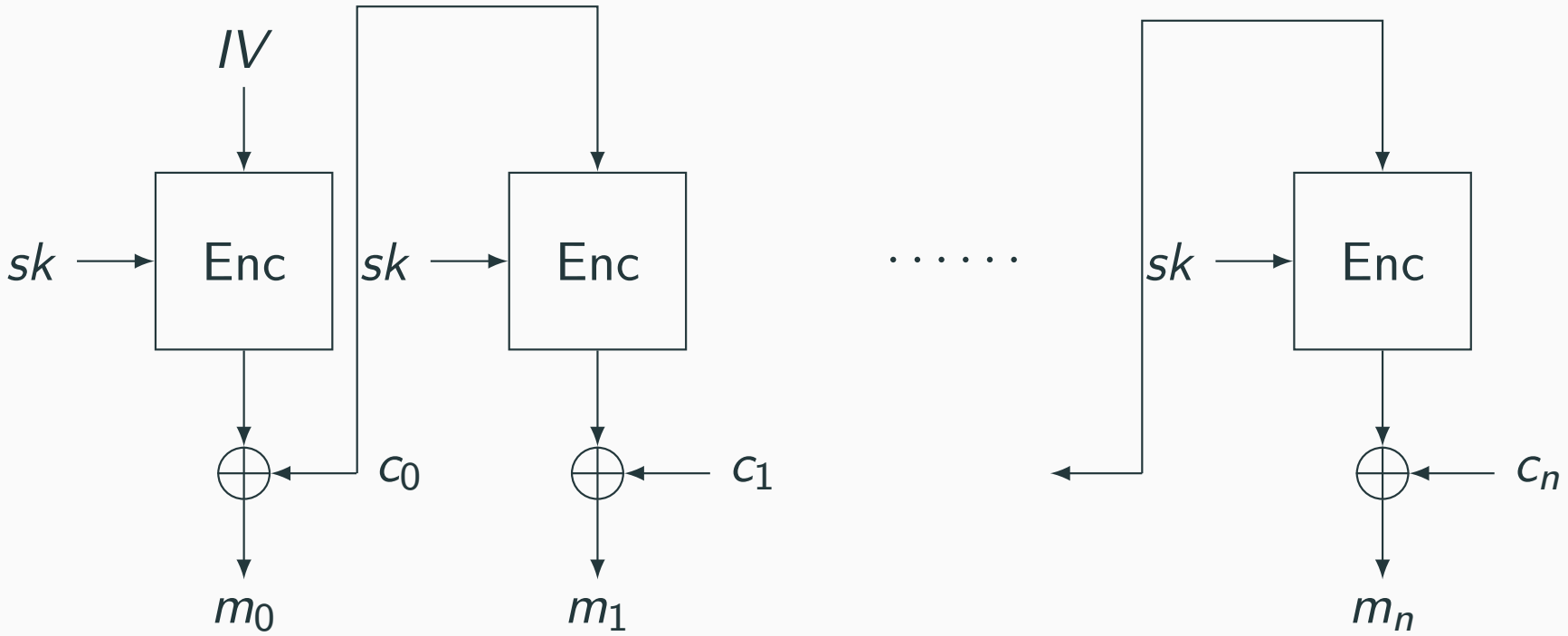
Mode CFB (Cipher FeedBack)

Encryption:



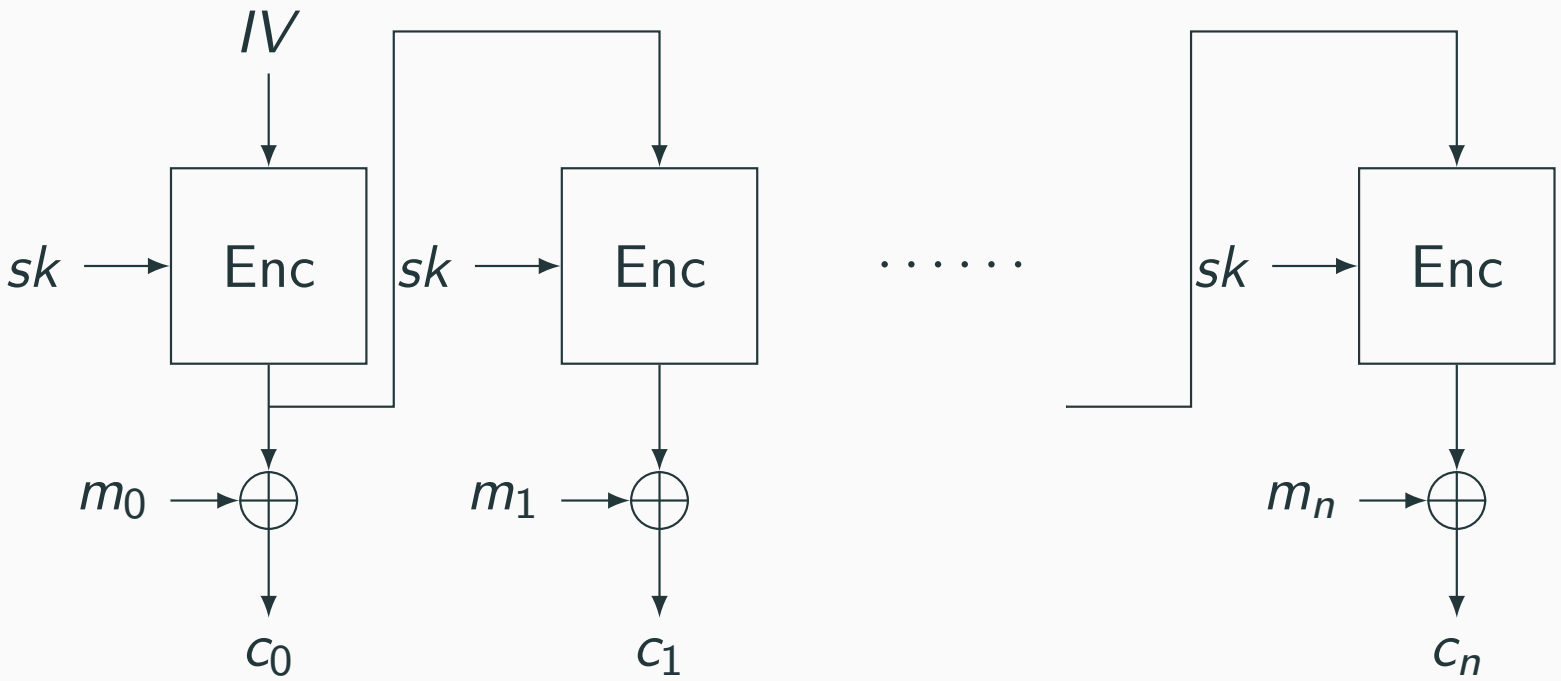
Mode CFB (Cipher FeedBack)

Decryption:



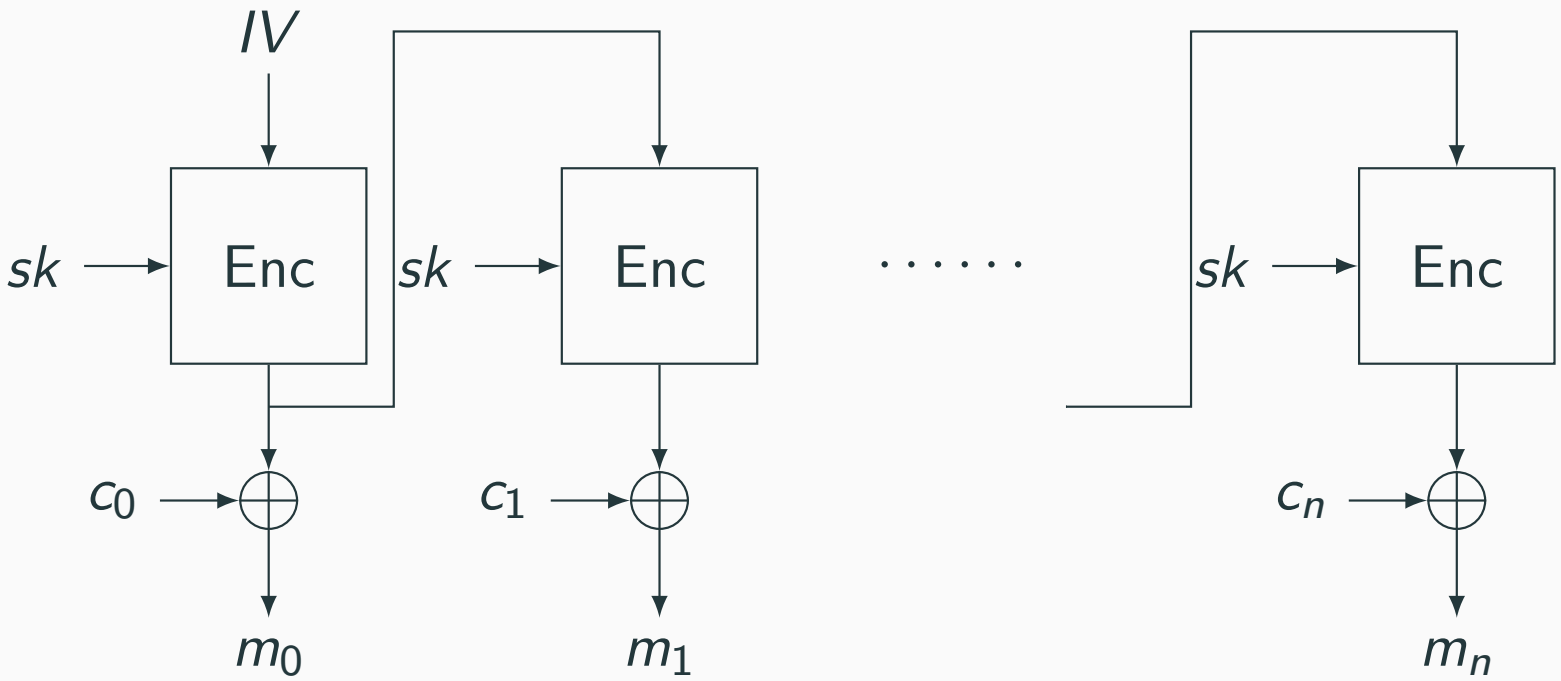
Mode OFB (Output FeedBack)

Encryption:



Mode OFB (Output FeedBack)

Decryption:



Note on CFB and OFB

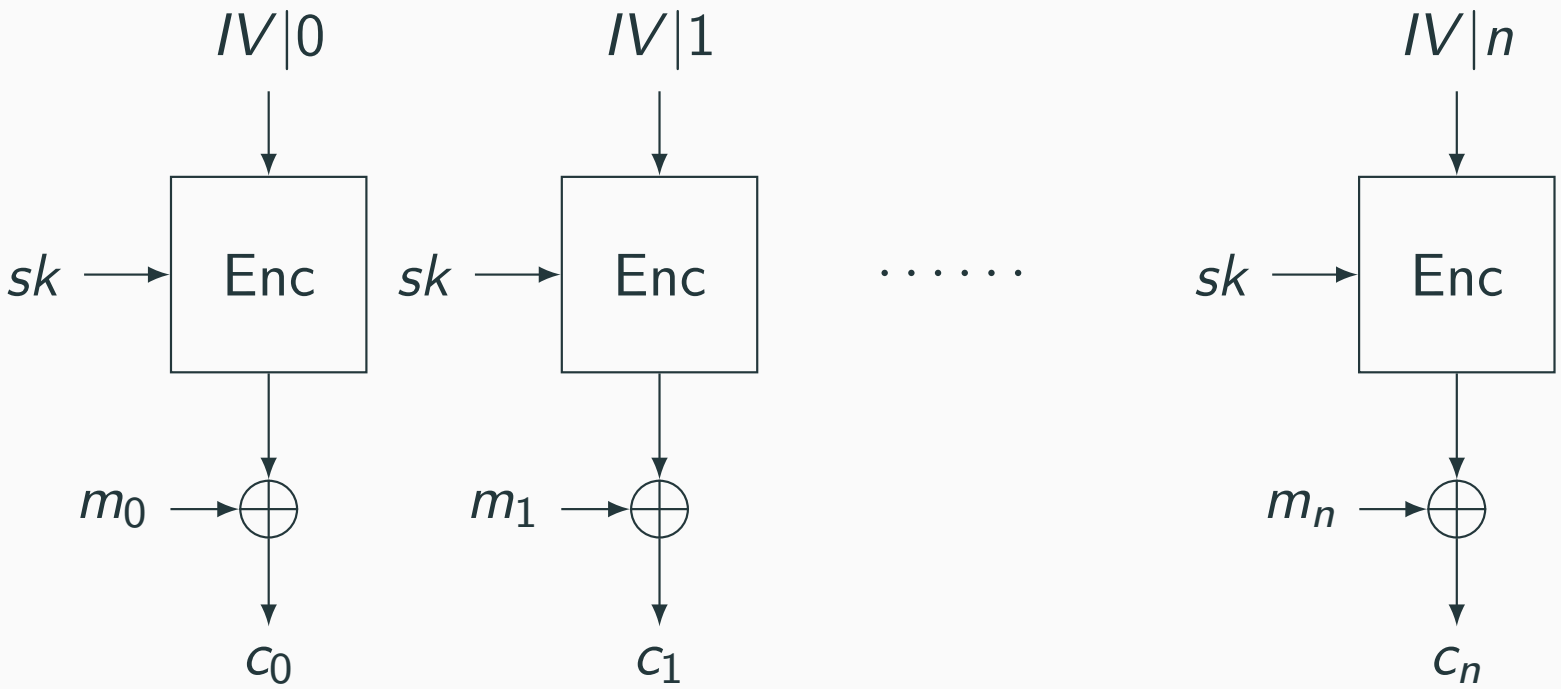
Note that for decryption, we use the Enc block again!

Why is that?

Note that CFB decryption can be parallelized.

Mode CTR (counter)

Encryption:



Notes on CTR mode

CTR mode can be parallelized.

Secure as long as Enc is secure (i.e, secure if used with AES, unsecure if used with DES), and the IV is not reused!

Which mode to use? (there are other mode as well but less popular)

There are little differences, but nothing critical. As long as you do not use ECB mode and that you do not reuse your IVs, you are fine.

Ferguson and Schneier recommend CBC and CTR.

Computational cost of encryption

2 hours of video (assumes 3Ghz CPU)

	DVD 4,7 G.B		Blu-Ray 25 GB	
Schemes	encrypt	decrypt	encrypt	decrypt
RSA 2048(1)	22 min	24 h	115 min	130 h
RSA 1024(1)	21 min	10 h	111 min	53 h
AES CTR(2)	20 sec	20 sec	105 sec	105 sec

Conclusion

When you read that such data is encrypted with AES-128-CBC, you can thus conclude that:

- The cryptosystem is AES
- AES key size is 128 bits
- Block chaining mode is CBC.

Important notions

- Difference between computational security and perfect secrecy
- Bits of security
- Shannon principles
- Feistel Networks
- Brief DES overview, DES weakness, 3DES
- Brief AES overview
- Each cipher operates on a fixed data size
- What are ECB, CBC, etc. How they work.
- Role of the nonce (IV), why it must never be reused

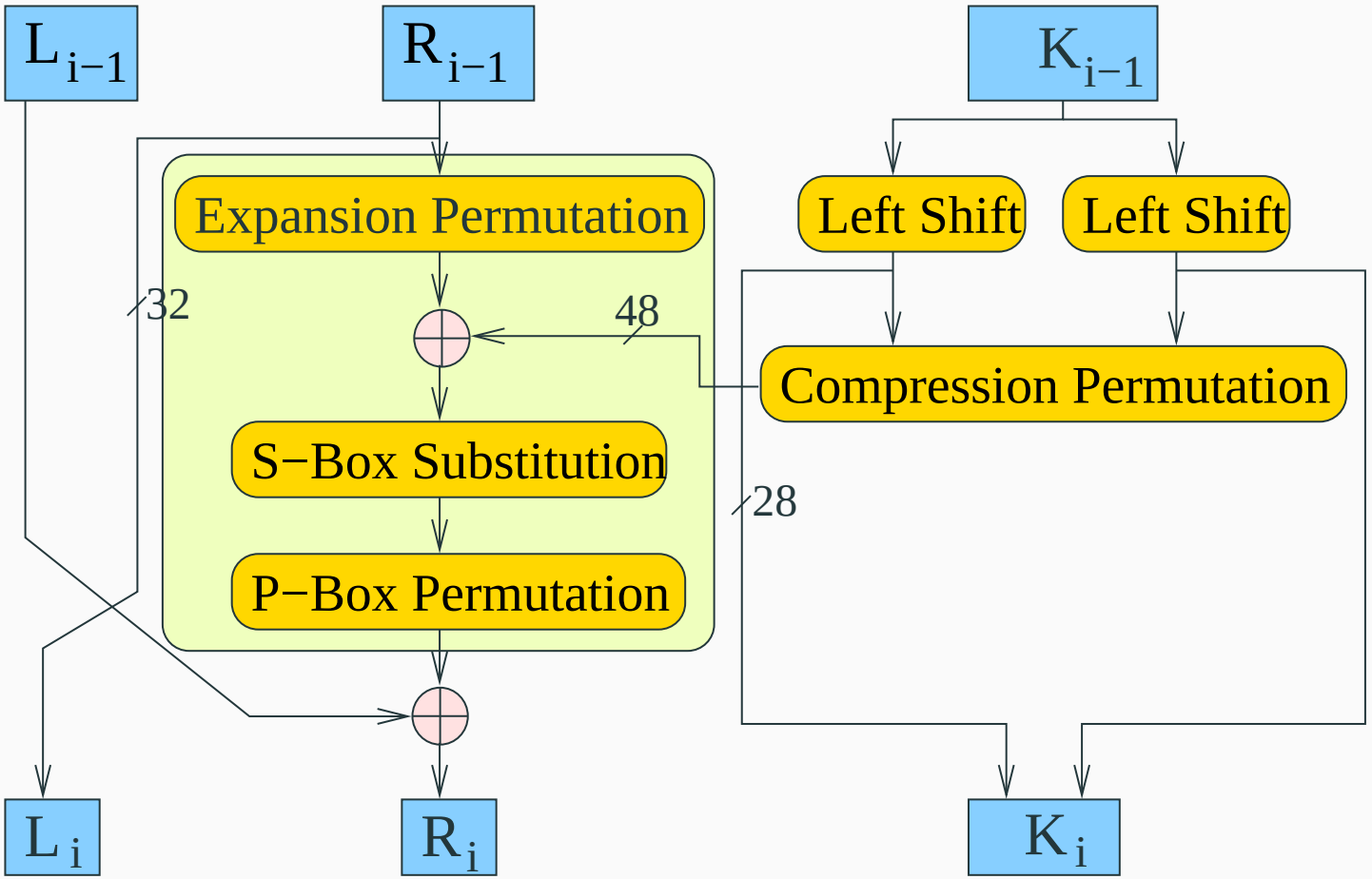
Exercises

1. On which technology does DES rely? What about AES?
2. Be able to use an S-box
3. Why is DES obsolete? What about 3DES?
4. Take one chaining mode (CBC...), draw the encryption scheme. Then find by yourself how the decryption works.

Appendix

This is just for your personal culture, if you want to know all the ugly details of DES and AES.

DES (Data Encryption Standard)



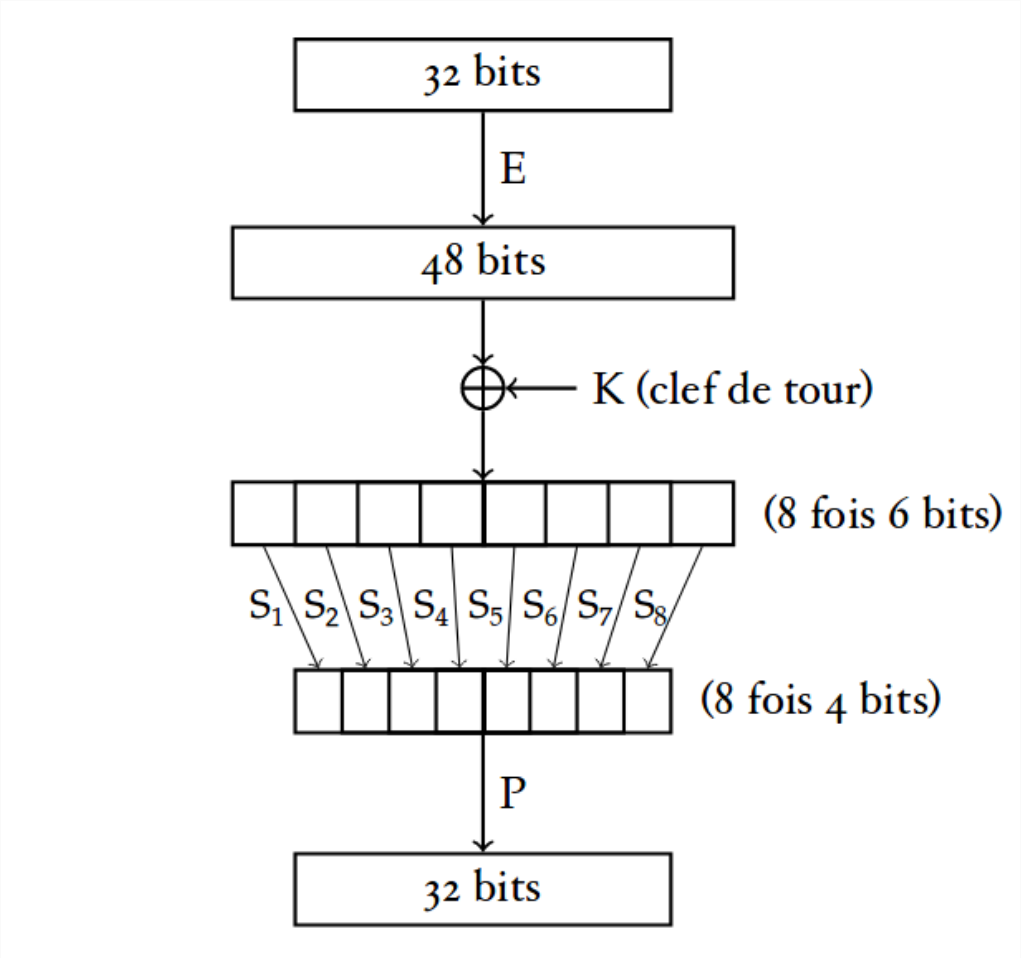
DES / Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

DES / Final Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

DES (Data Encryption Standard): Key expansion



DES / Expansion Permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

DES (Data Encryption Standard) - S-boxes

S-boxes
They are nonlinear boolean vectorial functions. They add *confusion*: the goal is to complexify the relation between the bits of the ciphertext and the key.

DES (Data Encryption Standard) - S-boxes

How do they work?

- They take 6 bits as input ($b_1 b_2 b_3 b_4 b_5 b_6$), and output 4 bits.
- We differentiate external bits ($b_1 b_6$) from internal bits ($b_2 b_3 b_4 b_5$).
- The output is given by the corresponding S-box

S-box table

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																
1																
2																
3																

S-Boxes: S1, S2, S3, S4

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-Boxes: S5, S6, S7 and S8

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

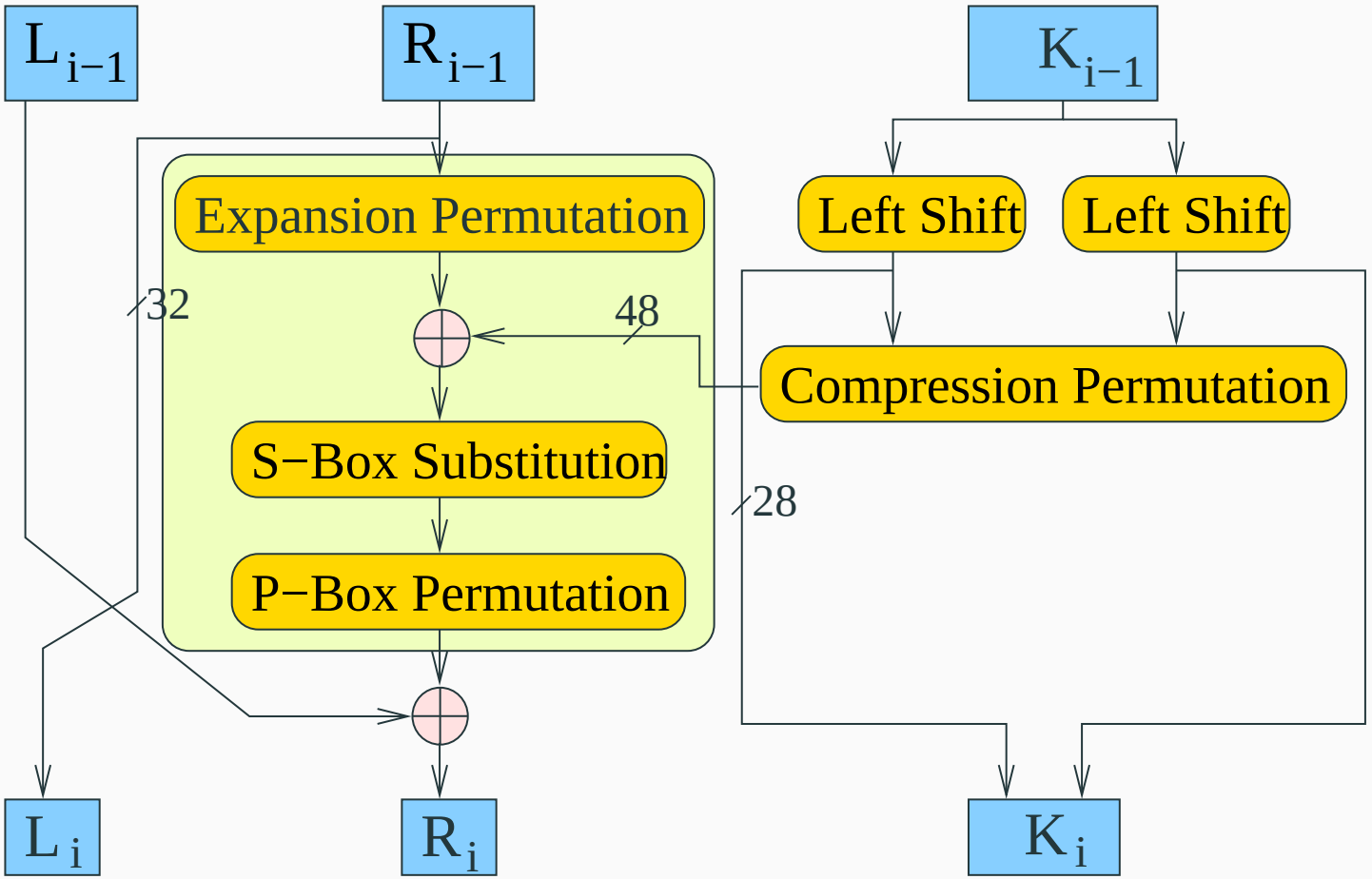
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

DES / Permutation (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

DES (Data Encryption Standard)



DES / Key Schedule

Input

- Secret key sk, $|sk| = 64$ bits
- Permutation Choice 1 (PC1)
- Permutation Choice 2 (PC2)
- Left shifts $r_1, r_2, \ldots, r_{16}$

Output

- 16 round keys: $k_1, k_2, \ldots, k_{16}$, $|k_i| = 48$ bits

DES / Permutation Choice (PC1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

DES / Permutation Choice (PC2)

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

DES / Left Shifts

r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}
1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES / Generation of k_1

1. Apply PC1 on sk to obtain k' of 56 bits
2. Divide k' in two 28-bit parts to obtain k'_1 and k'_2
3. Apply r_1 on k'_1 and k'_2 to obtain k''_1 and k''_2
4. Apply PC2 on $k''_1 \| k''_2$ to obtain k_1 of 48 bits.

DES / Generation of k_i ($2 \leq i \leq 16$)

1. Apply r_i on k_{i-1} to obtain k'_{i-1}
2. Apply PC2 on k'_{i-1} to obtain k_i of 48 bits.

AES: High-Level Cipher Algorithm

- KeyExpansion using Rijndael's key schedule
- Initial Round: AddRoundKey
- Rounds:
  1. SubBytes: a non-linear substitution step where each byte is replaced with another according to a lookup table.
  2. ShiftRows: a transposition step where each row of the state is shifted cyclically a certain number of steps.
  3. MixColumns: a mixing operation which operates on the columns of the state, combining the four bytes in each column
  4. AddRoundKey: each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule.
- Final Round (no MixColumns)
  1. SubBytes
  2. ShiftRows
  3. AddRoundKey

Key Schedule / Second Round

We have one of the four columns of the round key

Secret key

23	f1	45	02	25
6a	a5	e1	1c	e9
4e	c9	ff	5d	7d
b3	01	22	b4	d5

For the three other columns

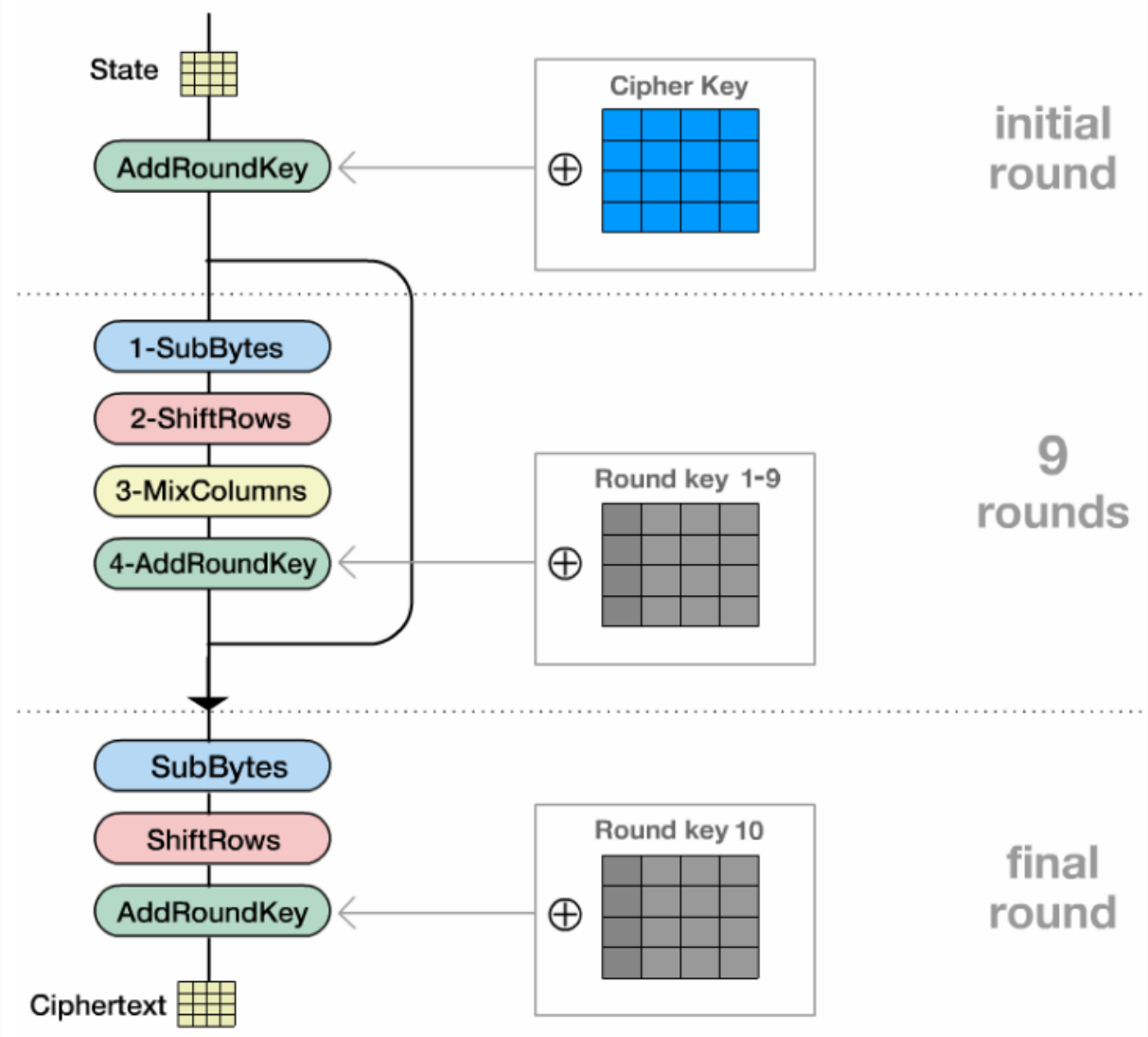
$$C_i = C_{i-1} \oplus C_{i-4}$$

with $6 \leq i \leq 8$

Key Schedule / Other Round

We repeat this process by replacing the secret key by the key of the second round, the key of the third round, and so on and so forth.

AES


State
AddRoundKey
Cipher Key
initial round
1-SubBytes
2-ShiftRows
3-MixColumns
4-AddRoundKey
Round key 1-9
9 rounds
SubBytes
ShiftRows
AddRoundKey
Round key 10
final round
Ciphertext

Key Schedule / Second Round

Secret key

23	f1	45	02
6a	a5	e1	1c
4e	c9	ff	5d
b3	01	22	b4

1. We take the last column
2. We apply a **up** rotation
3. SubBytes
4. Add first column and Rcon(1)

Key Schedule / Second Round

Secret key

23	f1	45	02
6a	a5	e1	1c
4e	c9	ff	5d
b3	01	22	b4

1. We take the last column
2. We apply a **up** rotation
3. SubBytes
4. Add first column and Rcon(1)

02	$\Rightarrow$	1c	$\Rightarrow$	9c	$\Rightarrow$	25
1c		5d		4c		e9
5d		b4		8d		7d
b4		02		77		d5

Key Schedule / Second Round

Secret key

23	f1	45	02
6a	a5	e1	1c
4e	c9	ff	5d
b3	01	22	b4

1. We take the last column
2. We apply a **up** rotation
3. SubBytes
4. Add first column and Rcon(1)

02		1c		9c		25
1c	$\Rightarrow$	5d	$\Rightarrow$	4c	$\Rightarrow$	e9
5d		b4		8d		7d
b4		02		77		d5

This column becomes the first column of the second round key